Collective Communication Routines

Scope:

- Collective communication routines must involve **all** processes within the scope of a communicator.
  - All processes are by default, members in the communicator MPI_COMM_WORLD.
  - Additional communicators can be defined by the programmer. See the Group and Communicator Management Routines section for details.

- Unexpected behavior, including program failure, can occur if even one task in the communicator doesn't participate.

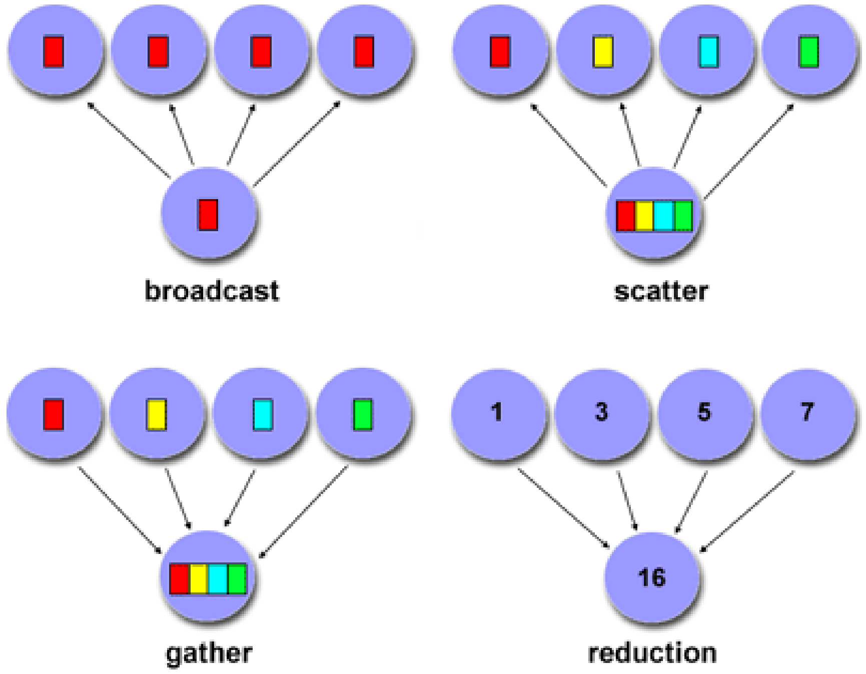
- It is the programmer's responsibility to ensure that all processes within a communicator participate in any collective operations.

Types of Collective Operations:

- **Synchronization** - processes wait until all members of the group have reached the synchronization point.

- **Data Movement** - broadcast, scatter/gather, all to all.

- **Collective Computation** (reductions) - one member of the group collects data from the other members and performs an operation (min, max, add, multiply, etc.) on that data.



Programming Considerations and Restrictions:

- Collective communication routines do not take message tag arguments.

- Collective operations within subsets of processes are accomplished by first partitioning the subsets into new groups and then attaching the new groups to new communicators (discussed in the Group and Communicator Management Routines section).

- Can only be used with MPI predefined datatypes - not with MPI Derived Data Types.

- MPI-2 extended most collective operations to allow data movement between intercommunicators (not covered here).

- With MPI-3, collective operations can be blocking or non-blocking. Only blocking operations are covered in this tutorial.

Collective Communication Routines

MPI_Barrier

Synchronization operation. Creates a barrier synchronization in a group. Each task, when reaching the MPI_Barrier call, blocks until all tasks in the group reach the same MPI_Barrier call. Then all tasks are free to proceed.

```
MPI_Barrier (comm)
MPI_BARRIER (comm,ierr)
```

MPI_Bcast

Data movement operation. Broadcasts (sends) a message from the process with rank "root" to all other processes in the group.

Diagram Here

```
MPI_Bcast (&buffer,count,datatype,root,comm)
MPI_BCAST (buffer,count,datatype,root,comm,ierr)
```

MPI_Scatter

Data movement operation. Distributes distinct messages from a single source task to each task in the group.

Diagram Here

```
MPI_Scatter (&sendbuf,sendcnt,sendtype,&recvbuf,
...... recvcnt,recvtype,root,comm)
MPI_SCATTER (sendbuf,sendcnt,sendtype,recvbuf,
...... recvcnt,recvtype,root,comm,ierr)
```

MPI_Gather

Data movement operation. Gathers distinct messages from each task in the group to a single destination task. This routine is the reverse operation of MPI_Scatter.

Diagram Here

```
MPI_Gather (&sendbuf,sendcnt,sendtype,&recvbuf,
...... recvcount,recvtype,root,comm)
MPI_GATHER (sendbuf,sendcnt,sendtype,recvbuf,
...... recvcount,recvtype,root,comm,ierr)
```

MPI_Allgather

Data movement operation. Concatenation of data to all tasks in a group. Each task in the group, in effect, performs a one-to-all broadcasting operation within the group.

Diagram Here

```
MPI_Allgather (&sendbuf,sendcount,sendtype,&recvbuf,
...... recvcount,recvtype,comm)
MPI_ALLGATHER (sendbuf,sendcount,sendtype,recvbuf,
...... recvcount,recvtype,comm,info)
```

MPI_Reduce

Collective computation operation. Applies a reduction operation on all tasks in the group and places the result in one task.

Diagram Here

```
MPI_Reduce (&sendbuf,&recvbuf,count,datatype,op,root,comm)
MPI_REDUCE (sendbuf,recvbuf,count,datatype,op,root,comm,ierr)
```

The predefined MPI reduction operations appear below. Users can also define their own reduction functions by using the MPI_Op_create routine.

MPI Reduction Operation		C Data Types	Fortran Data Type
MPI_MAX	maximum	integer, float	integer, real, complex
MPI_MIN	minimum	integer, float	integer, real, complex
MPI_SUM	sum	integer, float	integer, real, complex
MPI_PROD	product	integer, float	integer, real, complex
MPI_LAND	logical AND	integer	logical
MPI_BAND	bit-wise AND	integer, MPI_BYTE	integer, MPI_BYTE
MPI_LOR	logical OR	integer	logical
MPI_BOR	bit-wise OR	integer, MPI_BYTE	integer, MPI_BYTE
MPI_LXOR	logical XOR	integer	logical
MPI_BXOR	bit-wise XOR	integer, MPI_BYTE	integer, MPI_BYTE
MPI_MAXLOC	max value and location	float, double and long double	real, complex,double precision
MPI_MINLOC	min value and location	float, double and long double	real, complex, double precision

MPI_Allreduce

Collective computation operation + data movement. Applies a reduction operation and places the result in all tasks in the group. This is equivalent to an MPI_Reduce followed by an MPI_Bcast.

Diagram Here

```
MPI_Allreduce (&sendbuf,&recvbuf,count,datatype,op,comm)
MPI_ALLREDUCE (sendbuf,recvbuf,count,datatype,op,comm,ierr)
```

MPI_Reduce_scatter

Collective computation operation + data movement. First does an element-wise reduction on a vector across all tasks in the group. Next, the result vector is split into disjoint segments and distributed across the tasks. This is equivalent to an MPI_Reduce followed by an MPI_Scatter operation.

Diagram Here

```
MPI_Reduce_scatter (&sendbuf,&recvbuf,recvcount,datatype,
...... op,comm)
MPI_REDUCE_SCATTER (sendbuf,recvbuf,recvcount,datatype,
...... op,comm,ierr)
```

MPI_Alltoall

Data movement operation. Each task in a group performs a scatter operation, sending a distinct message to all the tasks in the group in order by index.

Diagram Here

```
MPI_Alltoall (&sendbuf,sendcount,sendtype,&recvbuf,
...... recvcnt,recvtype,comm)
MPI_ALLTOALL (sendbuf,sendcount,sendtype,recvbuf,
...... recvcnt,recvtype,comm,ierr)
```

MPI_Scan

Performs a scan operation with respect to a reduction operation across a task group.

Diagram Here

```
MPI_Scan (&sendbuf,&recvbuf,count,datatype,op,comm)
MPI_SCAN (sendbuf,recvbuf,count,datatype,op,comm,ierr)
```